UNIVERSIDADE SÃO FRANCISCO

Curso de Análise e Desenvolvimento de Sistemas - EAD

Nome	RA
Gabriel Faccina	202408991
Eduardo Souza Rodrigues	202441573
José Nadson Guedes Da Silva	202427732
Felipe Carlos De Oliveira Alves	202427472
Nelly Benites Soares	202420303
Vinicius Hofman Alves	202415705

PRÁTICA PROFISSIONAL: FERRAMENTAS E TÉCNICAS DE PROGRAMAÇÃO - ENTREGA FINAL

TEMA DO PROJETO: REDUÇÃO DO DESPERDÍCIO DE ALIMENTOS EM MERCADOS

Campinas

2024

AGRADECIMENTOS

Gostaríamos de expressar nossa sincera gratidão a todos que contribuíram para a conclusão deste projeto. Agradecemos à Universidade São Francisco por proporcionar o ambiente de aprendizado e apoio contínuo em nossa jornada acadêmica. Também estendemos nosso agradecimento aos professores do curso de Análise e Desenvolvimento de Sistemas pelas orientações essenciais e incentivo constante ao longo do processo.

Reconhecemos a dedicação e o esforço de todos os membros da equipe – Gabriel Faccina, Eduardo Souza Rodrigues, José Nadson Guedes Da Silva, Felipe Carlos De Oliveira Alves, Nelly Benites Soares e Vinicius Hofman Alves – que trabalharam de forma colaborativa para o sucesso deste projeto. Por fim, somos gratos às nossas famílias e amigos pelo apoio incondicional durante todo o processo.

Muito obrigado a todos que, de alguma forma, contribuíram para tornar este trabalho uma realidade!

RESUMO

O projeto desenvolvido visa a criação de um sistema de gerenciamento de inventário para mercados, com o intuito de reduzir o desperdício de alimentos. Utilizando a linguagem C, o sistema implementa funções essenciais para a manipulação de dados, como adição, listagem, edição, exclusão e salvamento de itens, além de possibilitar a recuperação dos dados em futuras execuções.

O programa emprega alocação dinâmica de memória para garantir escalabilidade, validação de entradas de dados para evitar erros e uma interface simples de navegação. A estrutura de dados utilizada armazena informações detalhadas dos produtos, como ID, nome, quantidade, preço, data de validade e categoria, contribuindo para uma gestão eficiente e otimização do controle de estoque, visando a redução de desperdícios.

ABSTRACT

This project aims to create an inventory management system for markets, with the goal of reducing food waste. Using the C programming language, the system implements essential functions for data manipulation, such as adding, listing, editing, deleting, and saving items, as well as enabling data retrieval in future executions.

The program employs dynamic memory allocation to ensure scalability, data entry validation to prevent errors, and noa simple navigation interface. The data structure used stores detailed product information, including ID, name, quantity, price, expiration date, and category, contributing to efficient management and optimization of stock control, aiming to reduce waste.

SUMÁRIO

1. FINIÇÃO DO CÓDIGO

Explicação primeiro trecho de código:

- Estrutura Item: é uma estrutura que define o item, que tem cada item presente no inventário. Esta estrutura tem os campos a seguir:
- ID: é um inteiro que armazena o identificador do item.
- Nome: é uma string com até 100 caracteres que guarda o nome do produto.
- Quantidade: é um inteiro que guarda a quantidade em estoque.
- Preço: é um decimal para o preço unitário do produto.
- Data de validade: uma string de 10 caracteres no formato dd/mm/aaaa.
- Categoria: é uma string com até 50 caracteres que guarda a categoria do produto.
- Função adicionarItem:

  o Itens: um ponteiro para o array de itens.

  o Contador: é um ponteiro para o número atual no inventário.

  o Capacidade: é um ponteiro para a capacidade máxima do array.

Passos executados:

- Capacidade de Verificação: a função verifica o número de itens (contador) para ver se possui atingido a capacidade total (capacidade). Se sim, o seu valor dobrará o valor de *capacidade e realocará a memória de modo a caber o dobro de itens. Para isso, é usada a função realloc. Caso o realloc retorne um ponteiro nulo, ele alocará uma mensagem de erro e sairá do programa.
- Entradas de Dados: são entrados e verificados todos os campos do novo item;
- ID: lido um número inteiro e garantida a ser válido. Caso contrário, imprimirá uma mensagem e solicitará novamente:
- Nome: pega a string do nome por scanf.
- Quantidade: lê um número inteiro para quantidade e verifica a entrada.
- Preço: lê número decimal para o preço e verifica.
- Data de validade: pega a string da data por scanf com o formato dd/mm/aaaa.
- Categoria = lida a categoria do produto como string:

- Incremento do Contador: após ter adicionado com sucesso o item, o *contador é incrementado para mostrar o número de itens como atual no inventário. Por fim, é impresso mensagem se o item foi adicionado com sucesso.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

// Define a estrutura de dados para armazenar informações de cada item do inventário
// Cada item possui um identificador único, nome, quantidade em estoque, preco unitário,
// data de validade e categoria do produto
typedef struct {
    int id;
    char nome[100];         // Nome do produto com até 100 caracteres
    int quantidade;         // Quantidade disponível em estoque
    float preco;           // Preço unitário do produto
    char dataDeValidade[11]; // Data de validade no formato dd/mm/aaaa
    char categoria[50];     // Categoria do produto com até 50 caracteres
} Item;

// Função responsável por adicionar um novo item ao inventário
// Parâmetros:
// - itens: ponteiro para o array de itens
// - contador: número atual de itens no inventário
// - capacidade: capacidade máxima atual do array
void adicionarItem(Item **itens, int *contador, int *capacidade) {
    // Verifica se é necessário aumentar a capacidade do array
    if (*contador == *capacidade) {
        *capacidade *= 2;  // Dobra a capacidade do array
        *itens = (Item*)realloc(*itens, *capacidade * sizeof(Item));
        if (*itens == NULL) {
            printf("Erro ao alocar memoria!\n");
            exit(1);
        }
    }

    // Solicita e valida o ID do novo item
```

```c
    printf("Digite o ID: ");
    while (scanf("%d", &(*itens)[*contador].id) != 1) {
        printf("Entrada invalida. Digite o ID novamente: ");
        while (getchar() != '\n');
    }

    // Solicita o nome do item
    printf("Digite o Nome: ");
    scanf(" %[^\n]", (*itens)[*contador].nome);

    // Solicita e valida a quantidade
    printf("Digite a Quantidade: ");
    while (scanf("%d", &(*itens)[*contador].quantidade) != 1) {
        printf("Entrada invalida. Digite a Quantidade novamente: ");
        while (getchar() != '\n');
    }

    // Solicita e valida o preco
    printf("Digite o Preco: ");
    while (scanf("%f", &(*itens)[*contador].preco) != 1) {
        printf("Entrada invalida. Digite o Preco novamente: ");
        while (getchar() != '\n');
    }

    // Solicita a data de validade
    printf("Digite a Data de Validade (dd/mm/aaaa): ");
    scanf(" %[^\n]", (*itens)[*contador].dataDeValidade);

    // Solicita a categoria
    printf("Digite a Categoria: ");
    scanf(" %[^\n]", (*itens)[*contador].categoria);

    // Incrementa o contador de itens e exibe mensagem de sucesso
    (*contador)++;
    printf("Item adicionado com sucesso!\n");
}
```

1.1 FUNÇÃO “listarItens”

Propósito: Exibir todos os itens cadastrados no inventário.

- Parâmetros:
  - Item *itens: Um array de itens do inventário.
  - int contador: O número atual de itens no inventário.
- Descrição do funcionamento:
  - Condição de Itens Vazio: Se contador é zero, exibe uma mensagem informando que nenhum item foi registrado.
  - Seção de Cabeçalho: Exibe um cabeçalho formatado com os campos (ID, Nome, Quantidade, Preço, Validade, Categoria).
  - Estrutura da Tabela: Usa um loop for para percorrer cada item no array itens, exibindo os detalhes formatados em tabela.

```c

// Função para exibir todos os itens cadastrados no inventário
// Parâmetros:
// - itens: array de itens
// - contador: número atual de itens no inventário
void listarItens(Item *itens, int contador) {
    if (contador == 0) {
        printf("\nNenhum item registrado.\n");
    } else {
        // Exibe cabeçalho da tabela
        printf("\n%-5s %-30s %-10s %-10s %-12s %-20s\n", "ID", "Nome", "Quantidade", "Preco", "Validade", "Categoria");
        printf("-------------------------------------------------------------------------------------\n");
        // Lista todos os itens formatados em tabela
        for (int i = 0; i < contador; i++) {
            printf("%-5d %-30s %-10d %-10.2f %-12s %-20s\n",
                itens[i].id, itens[i].nome, itens[i].quantidade, itens[i].preco, itens[i].dataDeValidade, itens[i].categoria);
        }
    }
}

```

1.2 FUNÇÃO “editarItem”

Propósito: Editar os dados de um item existente no inventário, procurando pelo ID.

- Parâmetros:
  - Item *itens: Um array de itens do inventário.
  - int contador: Número de itens no inventário.
  - int id: O ID do item a ser editado.
- Descrição do funcionamento:
  - Busca pelo ID: Procura o item com o ID fornecido. Se encontrado, permite a edição dos dados.
  - Entrada dos Novos Dados: Solicita novos valores (nome, quantidade, preço, validade, categoria) e substitui as informações antigas.
  - Validação de Entrada: Valida os dados de quantidade e preço para evitar entradas inválidas.
  - Mensagem de Confirmação: Exibe uma mensagem de sucesso ao atualizar o item ou uma mensagem informando que o item com o ID fornecido não foi encontrado.

```c
// Função para editar um item existente no inventário
// Parâmetros:
// - itens: array de itens
// - contador: número atual de itens
// - id: identificador do item a ser editado
void editarItem(Item *itens, int contador, int id) {
    // Procura o item pelo ID
    for (int i = 0; i < contador; i++) {
        if (itens[i].id == id) {
            // Solicita novos dados para o item
            printf("Digite o novo Nome: ");
            scanf(" %[^\n]", itens[i].nome);

            printf("Digite a nova Quantidade: ");
            while (scanf("%d", &itens[i].quantidade) != 1) {
                printf("Entrada invalida. Digite a nova Quantidade novamente: ");
                while (getchar() != '\n');
            }

            printf("Digite o novo Preco: ");
            while (scanf("%f", &itens[i].preco) != 1) {
                printf("Entrada invalida. Digite o novo Preço novamente: ");
                while (getchar() != '\n');
            }

            printf("Digite a nova Data de Validade (dd/mm/aaaa): ");
            scanf(" %[^\n]", itens[i].dataDeValidade);

            printf("Digite a nova Categoria: ");
            scanf(" %[^\n]", itens[i].categoria);

            printf("Item atualizado com sucesso!\n");
            return;
        }
    }
    printf("Item com ID %d nao encontrado.\n", id);
}
```

1.3 FUNÇÃO “excluirItem”

Propósito: Remover um item do inventário, mantendo a organização do array.

- Parâmetros:
  - Item *itens: Um array de itens do inventário.
  - int *contador: Um ponteiro para o contador que armazena o número atual de itens.
  - int id: O ID do item a ser excluído.
- Descrição do funcionamento:
  - Busca pelo ID: Localiza o item com o ID fornecido.
  - Remoção do Item: Move os itens subsequentes uma posição para trás, eliminando o item desejado.
  - Atualização do Contador: Reduz o contador para refletir a remoção.
  - Confirmação: Exibe uma mensagem de sucesso ou informa que o item não foi encontrado.

```c
// Função para excluir um item do inventário
// Parâmetros:
// - itens: array de itens
// - contador: ponteiro para o número atual de itens
// - id: identificador do item a ser excluído
void excluirItem(Item *itens, int *contador, int id) {
    int i;
    // Procura o item pelo ID
    for (i = 0; i < *contador; i++) {
        if (itens[i].id == id) {
            break;
        }
    }

    // Se encontrou o item, realiza a exclusão
    if (i < *contador) {
        // Move todos os itens posteriores uma posição para trás
        for (int j = i; j < *contador - 1; j++) {
            itens[j] = itens[j + 1];
        }
        (*contador)--;
        printf("Item excluido com sucesso.\n");
    } else {
        printf("Item com ID %d nao encontrado.\n", id);
    }
}

```

1.4 FUNÇÃO “salvarDados”

Propósito: Salvar os itens do inventário em um arquivo binário.

- Parâmetros:
  - Item *itens: Um array de itens do inventário.
  - int contador: Número atual de itens no inventário.
- Descrição do funcionamento:
  - Abertura do Arquivo: Abre o arquivo inventario.bin no modo de escrita binária.
  - Gravação do Contador: Salva o número total de itens (contador) no arquivo.
  - Gravação dos Itens: Escreve todos os itens do array itens no arquivo binário.
  - Fechamento e Confirmação: Fecha o arquivo e exibe uma mensagem confirmando o salvamento.

```c
// Função para salvar os dados do inventário em arquivo binário
// Parâmetros:
// - itens: array de itens
// - contador: número atual de itens
void salvarDados(Item *itens, int contador) {
    FILE *arquivo = fopen("inventario.bin", "wb");
    if (arquivo == NULL) {
        printf("Erro ao abrir o arquivo para escrita!\n");
        return;
    }
    // Salva o contador e todos os itens no arquivo
    fwrite(&contador, sizeof(int), 1, arquivo);
    fwrite(itens, sizeof(Item), contador, arquivo);
    fclose(arquivo);
    printf("Dados salvos com sucesso!\n");
}

```

1.5 FUNÇÃO “carregarDados”

Propósito :inventario.binpar

Detalhe :

- Parâmetros :
    1. Item **itens: Hum p
    2. int *contador:
    3. int *capacidade:
- Descrição do funcionamento :
    1. Abertura do arquivo :
        - O arquivo inventario.biné aberto em modo bi"rb").
        - Sim arquivo == NULL, uma
    2. Leitura do contador :
        - O número de itens é lido do arquivo usado freade armazenado em *contador.
    3. Alocação de memória :
        - A capacidade do array é definida como o número de itens carregados ou um valor padrão de 100, sendo *contadorzero.
        - mallocé usado para alocar memória para o array de itens. Se mallocfalhar, uma mensagem de erro é impressa e o programa é encerrado com exit(1).
    4. Leitura dos itens :
        - Os itens são lidos do arquivo e armazenados no array alocado usando fread.
    5. Fechamento do arquivo :
        - O arquivo é fechado com fclosee uma mensagem de sucesso é impressa.

```c
// Função para carregar os dados do inventário do arquivo binário
// Parâmetros:
// - itens: ponteiro para o array de itens
// - contador: ponteiro para o número de itens
// - capacidade: ponteiro para a capacidade do array
void carregarDados(Item **itens, int *contador, int *capacidade) {
    FILE *arquivo = fopen("inventario.bin", "rb");
    if (arquivo == NULL) {
        printf("Nenhum dado para carregar ou erro ao abrir o arquivo!\n");
        return;
    }
    // Lê o contador do arquivo
    fread(contador, sizeof(int), 1, arquivo);
    // Aloca memória necessária
    *capacidade = *contador > 0 ? *contador : 100;
    *itens = (Item*)malloc(*capacidade * sizeof(Item));
    if (*itens == NULL) {
        printf("Erro ao alocar memoria!\n");
        fclose(arquivo);
        exit(1);
    }
    // Lê todos os itens do arquivo
    fread(*itens, sizeof(Item), *contador, arquivo);
    fclose(arquivo);
    printf("Dados carregados com sucesso!\n");
}

```

1.6 FUNÇÃO “exibirCabecalho”

Propósito : Exibir um cabeçalho visual que indica o início do sistema de gestão de estoque.

Detalhe :

- Imprima uma linha separadora e um título centralizado que indica que o sistema de gerenciamento de estoque foi iniciado.
- É útil para dar uma apresentação visual clara ao usuário quando o programa é executado.

```c
// Função para exibir o cabeçalho do programa
void exibirCabecalho() {
    printf("\n====================================\n");
    printf("   SISTEMA DE GESTAO DE ESTOQUE   \n");
    printf("====================================\n");
}

```

1.7 FUNÇÃO “exibirMenu”

Propósito : Exibir o menu principal com as transações disponíveis para o usuário.

Detalhe :

- Imprima uma lista de operações que o usuário pode realizar, como adicionar, listar, editar, remover itens e salvar/encerrar.

- A função finaliza com uma solicitação para que o usuário insira a letra da operação desejada.

```
207  // Função para exibir o menu principal com as operações disponíveis
208 void exibirMenu() {
209      printf("\nOPERACOES DISPONIVEIS:\n");
210      printf("------------------------\n");
211      printf("[A] Adicionar novo item\n");
212      printf("[L] Listar todos os itens\n");
213      printf("[E] Editar item existente\n");
214      printf("[R] Remover item do estoque\n");
215      printf("[S] Salvar e encerrar\n");
216      printf("------------------------\n");
217      printf("Digite a letra da operacao desejada: ");
218  }
219
```

1.8 FUNÇÃO “obterEscolha”

Propósito : Ler a escolha do usuário e normalizá-la para seguranças.

Detalhe :

- Lê um caractere de entrada do usuário com scanfe remova qualquer espaço em branco à esquerda com " %c".
- Usa-se toupperpara converter o caráter lido em autoridades, garantindo que a entrada seja protegida de forma uniforme, independentemente do usuário digitar letras secretas ou minúsculas.
- Retorna a escolha normalizada para que possa ser usada em instruções de controle, como switchou if-else.

```
220  // Função para obter e normalizar a escolha do usuário
221 char obterEscolha() {
222      char escolha;
223      scanf(" %c", &escolha);
224      return toupper(escolha);
225  }
226
```

int capacidade = 100;

Item *itens = NULL;

int contador = 0;

capacidade: define o limite inicial de itens que o array pode armazenar.

itens: é um ponteiro que representa o array dinâmico onde os itens serão guardados.

contador: controla o número atual de itens no array.

```
227  // Função principal do programa
228 int main() {
229      // Inicialização das variáveis principais
230      int capacidade = 100;  // Capacidade inicial do array
231      Item *itens = NULL;    // Array dinâmico de itens
232      int contador = 0;      // Contador de itens atual
233
```

1.9 FUNÇÃO carregarDados(&itens, &contador, &capacidade);

A função carregarDados recupera os itens que foram salvos anteriormente, preenchendo o array itens e atualizando contador e capacidade com esses dados.

```
234      // Carrega dados salvos anteriormente
235      carregarDados(&itens, &contador, &capacidade);
236
237      // Loop principal do programa
238      char escolha;
239      do {
240          exibirCabecalho();
241          exibirMenu();
242          escolha = obterEscolha();
243
```

```
char escolha;
do {
   exibirCabecalho();
   exibirMenu();
   escolha = obterEscolha();
```

“escolha” armazena a opção escolhida pelo usuário.

O loop “do...while” permite ao usuário realizar várias operações (como adicionar, listar, editar e excluir itens) até que ele saia do programa.

```
237      // Loop principal do programa
238      char escolha;
239      do {
240          exibirCabecalho();
241          exibirMenu();
242          escolha = obterEscolha();
243
```

```
switch (escolha) {
```

O switch verifica a escolha do usuário e executa a ação correspondente para cada caso.

```
case 'A':
    printf("\n--- ADICIONANDO NOVO ITEM ---\n");
    adicionarItem(&itens, &contador, &capacidade);
    break;
```

Exibe uma mensagem para indicar o início do processo de adição.
A função “adicionarItem” insere um novo item no array itens, aumentando contador e redimensionando o array se “capacidade” for atingida.

```
case 'L':
    printf("\n--- LISTAGEM DE ITENS ---\n");
    listarItens(itens, contador);
    break;
```

Exibe uma mensagem para a listagem de itens.
Chama “listarItens” que mostra todos os itens atualmente armazenados, usando “contador” como limite.

```
case 'E': {
    int id;
    printf("\n--- EDITANDO ITEM ---\n");
    printf("Informe o ID do item a ser editado: ");
    while (scanf("%d", &id) != 1) {
        printf("Entrada inválida. Informe o ID novamente: ");
        while (getchar() != '\n');
    }
    editarItem(itens, contador, id);
    break;
}
```

Solicita ao usuário o ID do item que deseja editar.
O loop while verifica se o ID fornecido é válido.
A função “editarItem” altera as informações do item identificado pelo ID.

```
244         // Processa a escolha do usuário
245         switch(escolha) {
246             case 'A':
247                 printf("\n--- ADICIONANDO NOVO ITEM ---\n");
248                 adicionarItem(&itens, &contador, &capacidade);
249                 break;
250             case 'L':
251                 printf("\n--- LISTAGEM DE ITENS ---\n");
252                 listarItens(itens, contador);
253                 break;
254             case 'E': {
255                 int id;
256                 printf("\n--- EDITANDO ITEM ---\n");
257                 printf("Informe o ID do item a ser editado: ");
258                 while (scanf("%d", &id) != 1) {
259                     printf("Entrada invalida. Informe o ID novamente: ");
260                     while (getchar() != '\n');
261                 }
262                 editarItem(itens, contador, id);
263                 break;
264             }
```

```
case 'R': {
    int id;
    printf("\n--- REMOVENDO ITEM ---\n");
    printf("Informe o ID do item a ser removido: ");
    while (scanf("%d", &id) != 1) {
        printf("Entrada invalida. Informe o ID novamente: ");
        while (getchar() != '\n');
    }
    excluirItem(itens, &contador, id);
    break;
}
```

Solicita o ID do item a ser removido.
A função “excluirItem” remove o item do array e ajusta contador para refletir a exclusão.

```
case 'S':
    printf("\nSalvando dados e encerrando o programa...\n");
    salvarDados(itens, contador);
    break;
```

A função “salvarDados” grava o conteúdo de itens em um arquivo, para que seja recuperado em futuras execuções do programa.
Encerra o loop “do...while”.

```
default:
    printf("\nOpcao invalida! Por favor, escolha uma operação válida.\n");
```

Exibe uma mensagem se o usuário digitar uma opção que não corresponde a nenhum dos casos válidos.

```
            case 'R': {
                int id;
                printf("\n--- REMOVENDO ITEM ---\n");
                printf("Informe o ID do item a ser removido: ");
                while (scanf("%d", &id) != 1) {
                    printf("Entrada invalida. Informe o ID novamente: ");
                    while (getchar() != '\n');
                }
                excluirItem(itens, &contador, id);
                break;
            }
            case 'S':
                printf("\nSalvando dados e encerrando o programa...\n");
                salvarDados(itens, contador);
                break;
            default:
                printf("\nOpcao invalida! Por favor, escolha uma operacao valida.\n");
        }
```

```
if (escolha != 'S') {
    printf("\nPressione Enter para continuar...");
    while (getchar() != '\n');
    getchar();
}
```

Pausa a execução para que o usuário possa ler as mensagens antes de retornar ao menu.

```
        // Pausa para o usuário ler as mensagens
        if (escolha != 'S') {
            printf("\nPressione Enter para continuar...");
            while (getchar() != '\n');
            getchar();
        }

    } while(escolha != 'S');

```

```
free(itens);

return 0;
```

free(itens): libera a memória usada pelo array itens antes de encerrar o programa.

return 0: encerra o programa, indicando que ele foi executado com sucesso.

```
    // Libera a memória alocada antes de encerrar
    free(itens);
    return 0;
}
```

REFERÊNCIAS

CODE::BLOCKS. **Code::Blocks: Free C/C++** IDE. Disponível em: https://www.codeblocks.org/. Acesso em: 03 nov. 2024.

TODA MATÉRIA. **Normas ABNT para trabalhos acadêmicos**. Disponível em: https://www.todamateria.com.br/normas-abnt-trabalhos/. Acesso em: 9 nov. 2024.

APÊNDICE - CÓDIGO DO PROJETO

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

// Define a estrutura de dados para armazenar informações de cada item do
inventário
// Cada item possui um identificador único, nome, quantidade em estoque, preço
unitário,
// data de validade e categoria do produto
typedef struct {
    int id;
    char nome[100];        // Nome do produto com até 100 caracteres
    int quantidade;        // Quantidade disponível em estoque
    float preco;           // Preço unitário do produto
    char dataDeValidade[11]; // Data de validade no formato dd/mm/aaaa
    char categoria[50];    // Categoria do produto com até 50 caracteres
} Item;

// Função responsável por adicionar um novo item ao inventário
// Parâmetros:
// - itens: ponteiro para o array de itens
// - contador: número atual de itens no inventário
// - capacidade: capacidade máxima atual do array
void adicionarItem(Item **itens, int *contador, int *capacidade) {
    // Verifica se é necessário aumentar a capacidade do array
    if (*contador == *capacidade) {
        *capacidade *= 2;  // Dobra a capacidade do array
        *itens = (Item*)realloc(*itens, *capacidade * sizeof(Item));
        if (*itens == NULL) {
            printf("Erro ao alocar memoria!\n");
            exit(1);
        }
    }

    // Solicita e valida o ID do novo item
    printf("Digite o ID: ");
    while (scanf("%d", &(*itens)[*contador].id) != 1) {
        printf("Entrada invalida. Digite o ID novamente: ");
        while (getchar() != '\n');
    }

    // Solicita o nome do item
    printf("Digite o Nome: ");
    scanf(" %[^\n]", (*itens)[*contador].nome);

    // Solicita e valida a quantidade
    printf("Digite a Quantidade: ");
```

```
            while (scanf("%d", &(*itens)[*contador].quantidade) != 1) {
                printf("Entrada invalida. Digite a Quantidade novamente: ");
                while (getchar() != '\n');
            }

            // Solicita e valida o preço
            printf("Digite o Preco: ");
            while (scanf("%f", &(*itens)[*contador].preco) != 1) {
                printf("Entrada invalida. Digite o Preco novamente: ");
                while (getchar() != '\n');
            }

            // Solicita a data de validade
            printf("Digite a Data de Validade (dd/mm/aaaa): ");
            scanf(" %[^\n]", (*itens)[*contador].dataDeValidade);

            // Solicita a categoria
            printf("Digite a Categoria: ");
            scanf(" %[^\n]", (*itens)[*contador].categoria);

            // Incrementa o contador de itens e exibe mensagem de sucesso
            (*contador)++;
            printf("Item adicionado com sucesso!\n");
        }

        // Função para exibir todos os itens cadastrados no inventário
        // Parâmetros:
        // - itens: array de itens
        // - contador: número atual de itens no inventário
        void listarItens(Item *itens, int contador) {
            if (contador == 0) {
                printf("\nNenhum item registrado.\n");
            } else {
                // Exibe cabeçalho da tabela
                printf("\n%-5s %-30s %-10s %-10s %-12s %-20s\n", "ID", "Nome",
"Quantidade", "Preco", "Validade", "Categoria");
                printf("-------------------------------------------------------------------------------------\n");
                // Lista todos os itens formatados em tabela
                for (int i = 0; i < contador; i++) {
                    printf("%-5d %-30s %-10d %-10.2f %-12s %-20s\n",
                        itens[i].id, itens[i].nome, itens[i].quantidade, itens[i].preco,
itens[i].dataDeValidade, itens[i].categoria);
                }
            }
        }

        // Função para editar um item existente no inventário
        // Parâmetros:
        // - itens: array de itens
        // - contador: número atual de itens
```

```
// - id: identificador do item a ser editado
void editarItem(Item *itens, int contador, int id) {
    // Procura o item pelo ID
    for (int i = 0; i < contador; i++) {
        if (itens[i].id == id) {
            // Solicita novos dados para o item
            printf("Digite o novo Nome: ");
            scanf(" %[^\n]", itens[i].nome);

            printf("Digite a nova Quantidade: ");
            while (scanf("%d", &itens[i].quantidade) != 1) {
                printf("Entrada invalida. Digite a nova Quantidade novamente: ");
                while (getchar() != '\n');
            }

            printf("Digite o novo Preco: ");
            while (scanf("%f", &itens[i].preco) != 1) {
                printf("Entrada invalida. Digite o novo Preço novamente: ");
                while (getchar() != '\n');
            }

            printf("Digite a nova Data de Validade (dd/mm/aaaa): ");
            scanf(" %[^\n]", itens[i].dataDeValidade);

            printf("Digite a nova Categoria: ");
            scanf(" %[^\n]", itens[i].categoria);

            printf("Item atualizado com sucesso!\n");
            return;
        }
    }
    printf("Item com ID %d nao encontrado.\n", id);
}

// Função para excluir um item do inventário
// Parâmetros:
// - itens: array de itens
// - contador: ponteiro para o número atual de itens
// - id: identificador do item a ser excluído
void excluirItem(Item *itens, int *contador, int id) {
    int i;
    // Procura o item pelo ID
    for (i = 0; i < *contador; i++) {
        if (itens[i].id == id) {
            break;
        }
    }

    // Se encontrou o item, realiza a exclusão
    if (i < *contador) {
```

```
            // Move todos os itens posteriores uma posição para trás
            for (int j = i; j < *contador - 1; j++) {
                itens[j] = itens[j + 1];
            }
            (*contador)--;
            printf("Item excluido com sucesso.\n");
        } else {
            printf("Item com ID %d nao encontrado.\n", id);
        }
    }

    // Função para salvar os dados do inventário em arquivo binário
    // Parâmetros:
    // - itens: array de itens
    // - contador: número atual de itens
    void salvarDados(Item *itens, int contador) {
        FILE *arquivo = fopen("inventario.bin", "wb");
        if (arquivo == NULL) {
            printf("Erro ao abrir o arquivo para escrita!\n");
            return;
        }
        // Salva o contador e todos os itens no arquivo
        fwrite(&contador, sizeof(int), 1, arquivo);
        fwrite(itens, sizeof(Item), contador, arquivo);
        fclose(arquivo);
        printf("Dados salvos com sucesso!\n");
    }

    // Função para carregar os dados do inventário do arquivo binário
    // Parâmetros:
    // - itens: ponteiro para o array de itens
    // - contador: ponteiro para o número de itens
    // - capacidade: ponteiro para a capacidade do array
    void carregarDados(Item **itens, int *contador, int *capacidade) {
        FILE *arquivo = fopen("inventario.bin", "rb");
        if (arquivo == NULL) {
            printf("Nenhum dado para carregar ou erro ao abrir o arquivo!\n");
            return;
        }
        // Lê o contador do arquivo
        fread(contador, sizeof(int), 1, arquivo);
        // Aloca memória necessária
        *capacidade = *contador > 0 ? *contador : 100;
        *itens = (Item*)malloc(*capacidade * sizeof(Item));
        if (*itens == NULL) {
            printf("Erro ao alocar memoria!\n");
            fclose(arquivo);
            exit(1);
        }
        // Lê todos os itens do arquivo
```

```
    fread(*itens, sizeof(Item), *contador, arquivo);
    fclose(arquivo);
    printf("Dados carregados com sucesso!\n");
}

// Função para exibir o cabeçalho do programa
void exibirCabecalho() {
    printf("\n==================================\n");
    printf("   SISTEMA DE GESTAO DE ESTOQUE   \n");
    printf("==================================\n");
}

// Função para exibir o menu principal com as operações disponíveis
void exibirMenu() {
    printf("\nOPERACOES DISPONIVEIS:\n");
    printf("----------------------\n");
    printf("[A] Adicionar novo item\n");
    printf("[L] Listar todos os itens\n");
    printf("[E] Editar item existente\n");
    printf("[R] Remover item do estoque\n");
    printf("[S] Salvar e encerrar\n");
    printf("----------------------\n");
    printf("Digite a letra da operacao desejada: ");
}

// Função para obter e normalizar a escolha do usuário
char obterEscolha() {
    char escolha;
    scanf(" %c", &escolha);
    return toupper(escolha);
}

// Função principal do programa
int main() {
    // Inicialização das variáveis principais
    int capacidade = 100;  // Capacidade inicial do array
    Item *itens = NULL;    // Array dinâmico de itens
    int contador = 0;      // Contador de itens atual

    // Carrega dados salvos anteriormente
    carregarDados(&itens, &contador, &capacidade);

    // Loop principal do programa
    char escolha;
    do {
        exibirCabecalho();
        exibirMenu();
        escolha = obterEscolha();

        // Processa a escolha do usuário
```

```
    switch(escolha) {
        case 'A':
            printf("\n--- ADICIONANDO NOVO ITEM ---\n");
            adicionarItem(&itens, &contador, &capacidade);
            break;
        case 'L':
            printf("\n--- LISTAGEM DE ITENS ---\n");
            listarItens(itens, contador);
            break;
        case 'E': {
            int id;
            printf("\n--- EDITANDO ITEM ---\n");
            printf("Informe o ID do item a ser editado: ");
            while (scanf("%d", &id) != 1) {
                printf("Entrada invalida. Informe o ID novamente: ");
                while (getchar() != '\n');
            }
            editarItem(itens, contador, id);
            break;
        }
        case 'R': {
            int id;
            printf("\n--- REMOVENDO ITEM ---\n");
            printf("Informe o ID do item a ser removido: ");
            while (scanf("%d", &id) != 1) {
                printf("Entrada invalida. Informe o ID novamente: ");
                while (getchar() != '\n');
            }
            excluirItem(itens, &contador, id);
            break;
        }
        case 'S':
            printf("\nSalvando dados e encerrando o programa...\n");
            salvarDados(itens, contador);
            break;
        default:
            printf("\nOpcao invalida! Por favor, escolha uma operacao valida.\n");
    }

    // Pausa para o usuário ler as mensagens
    if (escolha != 'S') {
        printf("\nPressione Enter para continuar...");
        while (getchar() != '\n');
        getchar();
    }

} while(escolha != 'S');

// Libera a memória alocada antes de encerrar
free(itens);
```

```
    return 0;
}
```